Bild: Albert Hulm

Besuch im RESTaurant

Webdienste per REST-Schnittstelle anzapfen

Damit eine Anwendung mit anderen Anwendungen kommunizieren kann, müssen diese Programmierschnittstellen bereitstellen. Im Web sind viele dieser APIs nach dem Quasi-Standard REST (REpresentational State Transfer) aufgebaut. Wer das Prinzip einmal verstanden hat, kann leicht externe Dienste für eigene Anwendungen nutzen.

Von Manuel Ottlik

Startet der Benutzer in einem Hotelvergleichsportal eine Suche, beginnt der Server, Informationen von verschiedenen Hotelanbietern zusammenzusammeln. Er nutzt dafür Application Programming Interfaces, kurz APIs, die von den Anbietern bereitgestellt werden und strukturierte Daten liefern. Wie eine solche Schnittstelle aussieht, ist nicht per Gesetz geregelt – es gibt im Web schließlich keine autoritäre Instanz, die festlegt, wie ein API gestaltet sein muss. Das Web organisiert sich vielmehr über gemeinsam vereinbarte Standards. APIs nach dem REST-Modell aufzubauen ist eine solche Übereinkunft. Wer einen Dienst mit einer REST-Schnittstelle anbietet, stellt sicher, dass andere Entwickler ihn schnell in ihre Anwendungen integrieren können. Doch wie sieht eine Schnittstelle aus, die für den Nutzer meist im Verborgenen bleibt?

Der Kellner

Ein API ist wie ein Vertrag zwischen zwei Programmen: Er legt fest, wie die beiden Programme miteinander sprechen und gegenseitig Informationen austauschen. Vereinbart wird zum Beispiel, in welcher Struktur und Sprache eine Informationsanfrage und dessen Antwort formuliert sind.

Aber einmal anschaulich: So wie ein Kellner in einem Restaurant kann ein API Befehle entgegennehmen und diese an die Küche, also das Backend, weitergeben. Der Kunde wählt ein Gericht von der Speisekarte und bestellt es beim Kellner. Der Kellner prüft, ob das Gericht auf der Karte steht, und gibt die Bestellung an die Küche

weiter. Der Gast muss sich nicht damit auskennen, wie ein Schnitzel zubereitet wird und bekommt die Küche meistens nicht zu Gesicht. Er verlangt einfach nach einem Schnitzel. Nach der Bestellung kommt das zubereitete Gericht, die Antwort des Backends, zurück zum Anwender.

Ein Web-Frontend mit einem kleinen Bestellformular, das mit einem API arbeitet, macht das Gleiche: Per Klick auf „Speichern“ sendet der Browser die Daten an den Endpoint eines REST-APIs. Das Backend nimmt die Bestellung auf und speichert die Formulardaten in seiner Datenbank. Als Antwort gibt das Backend kurz Bescheid, ob das Speichern erfolgreich war.

Der Frack

Die Abkürzung REST und alle Übersetzungsversuche des „REpresentational State Transfer“ ergeben ein schiefes Bild. Die Übertragung geschieht nämlich gerade über zustandslose Protokolle wie HTTP. Um zu beschreiben, was ein REST-API tut, hilft die Abkürzung also nicht weiter: Das API bietet Schnittstellen an, um Informationen über ein Objekt zu senden oder anzufragen.

Bei den Objekten handelt es sich meist um Dinge aus der realen Welt. Als Beispiel soll in diesem Artikel eine Raumverwaltung dienen. Diese hat im einfachsten Fall zwei Objekte: Etagen und Räume. Wenn ein solches Objekt angefragt wird, sendet der Server ein Paket mit den Informationen zum Raum zurück. Diese können zum Beispiel aus einer oder mehreren Datenbanken kommen – das Zusammensammeln ist Aufgabe des Backends.

Die Speisekarte

Theoretisch kann man ein REST-API mit jedem zustandslosen Client-Server Protokoll realisieren. Um die gewohnte Benennung der Endpunkte von REST einzuhalten, führt jedoch kein Weg am Web-Protokoll HTTP vorbei. REST nutzt nämlich für die Endpunkte die HTTP-Methoden GET, POST, PUT, PATCH und DELETE. Dabei gilt die Prämisse, dass unter jedem Endpunkt eine Ressource anzufinden ist. Bei einer Raumverwaltung mit Etagen und Räumen gäbe es also zwei Ressourcen: `levels` und `rooms` – die Namen sollten immer im englischen Plural sein. Die Endpunkte für eine Ressource finden Sie in der Tabelle.

Genau die gleichen Endpunkte bekommt auch die Ressource `levels`. Da Räume eindeutig Etagen untergeordnet sind, empfiehlt es sich, auch dafür einen Endpunkt anzulegen. So wäre ein Endpoint sinnvoll, der mit dem Aufruf von `GET levels/{id}/rooms` eine Liste von allen Räumen aus der Etage mit der ID `{id}` zurückgibt. REST nutzt nicht nur die Methoden, sondern auch die Statuscodes von HTTP. Werden nicht existierende Endpoints (z. B. ein Raum mit der ID `-1`) angefragt, antwortet der Server mit dem bekannten Code `404 Not Found`. Wird eine Ressource erstellt, antwortet der Server mit `201 Created`. Die normale Datenabfrage mit GET wird bei Erfolg mit `200 OK` bestätigt – eine vollständige Auflistung aller HTTP-Statuscodes finden Sie über ct.de/yuz6.

Zur Adresse des APIs gehört auch ein Hostname. APIs sind oft als Subdomain oder Unterordner einer regulären Seite eingerichtet, also etwa unter `api.example.org` oder `example.org/api` zu erreichen. Wenn man das API als Vertrag zwischen zwei Programmen versteht, sollte dieser Vertrag nicht gebrochen werden – zum Beispiel durch eine Umbenennung der Endpunkte. Wenn sich grundlegende Strukturen ändern, hätten alle Nutzer des APIs große Probleme. Um dem aus dem Weg zu gehen und dennoch neue Funktionen anbieten zu können, werden APIs häufig versioniert und die Version in die Adresse aufgenommen. Die vollständige URL für den Endpunkt der Räume könnte also so aussehen: `example.org/api/v10/rooms`.

Das Gericht

Hinter den Endpoints verbergen sich Gerichte – sie kommen aus der Küche, also dem Backend. Ähnlich wie in einem Restaurant soll das Backend bis auf Weiteres eine „Blackbox“ bleiben, Betriebssystem und Programmiersprache sind nämlich völlig egal. Das Gericht kann sowohl von einer PHP-Anwendung auf Windows-Basis als auch einem Node.js-Framework auf Linux zubereitet werden.

In welcher Art und Weise die REST-Schnittstelle die Antworten zurückgibt, ist nicht festgelegt und muss dokumentiert werden. Grundsätzlich ist jede von Maschinen lesbare Auszeichnungssprache denkbar. Durchgesetzt hat sich jedoch vor allem JSON, die „JavaScript Object Notation“. Viele Web-Frontends sind in JavaScript geschrieben, sodass die Antworten direkt verwendet werden können. Anders

Endpunkte einer REST-Ressource

URL	HTTP-Methode	Funktion
/rooms	GET	gibt alle Datensätze zurück
/rooms	POST	erstellt einen neuen Datensatz
/rooms/{id}	GET	gibt den Datensatz mit der ID {id} zurück
/rooms/{id}	PUT	aktualisiert den kompletten Datensatz mit der ID {id}
/rooms/{id}	PATCH	aktualisiert Teile des Datensatzes mit der ID {id}
/rooms/{id}	DELETE	löscht den Datensatz mit der ID {id}

REST und andere Standards

REST-APIs gelten heutzutage nahezu synonym für sämtliche Web-APIs und werden von fast jedem Unternehmen eingesetzt. REST hat in den letzten Jahren andere Ansätze wie SOAP (Simple Object Access Protocol) weitgehend verdrängt. Genau genommen handelt es sich bei REST aber nur um eine Vereinbarung, nicht um eine komplette Schnittstelle. Das Muster legt fest, wie die Endpunkte des APIs gestaltet werden sollen, also unter welchen Routen welche Datensätze ausgelesen oder manipuliert werden können. Das lässt dem Entwickler viel Freiheit. Gleichzeitig geht die Auffassung über die Regeln, an die sich ein „richtiges“ REST-API zu halten hat, weit auseinander. Auch wenn sich mittlerweile ein weit verbreiteter Standard entwickelt hat, kommt jede gute API mit einer mindestens genauso guten Dokumentation. Sie gibt Aufschluss über die Struktur des APIs und listet alle Endpoints mit den zu erwartenden Rückgaben auf. Als Standard für die Dokumentation hat sich der OpenAPI-Standard etabliert (früher bekannt als „Swagger“). Die Dokumentation finden Sie über ct.de/yuz6.

```
{
 "meta": {
  "status": 200,
  "response_ms": 341,
  "items": 3
 },
 "payload": {
  "0": {
   "id": 1,
   "name": "Konferenzraum",
   "description": "Telefonspinne, Fensterfront",
   "seats": 18,
   "bookable": true,
   "level_id": 2,
   "links": [
    {"rel": "level", "href": "/levels/2"},
    {"rel": "book", "href": "/rooms/1/book"}
   ]
  },
  [...]
  //Zwei weitere Räume
 }
}
```

Der Server liefert die Daten über alle Räume zurück und liefert die Adressen anderer Endpunkte gleich mit.

als zum Beispiel XML hat JSON auch deutlich weniger Overhead. Das zurückgegebene Objekt besteht meist aus zwei Teilen: den Metadaten und der sogenannten Payload, also den angefragten Daten. Die Metadaten liefern oft den HTTP-Statuscode noch einmal mit, außerdem die benötigte Antwortzeit und die Anzahl der zurückgesendeten Objekte. Die Payload enthält die aufbereiteten Daten. Ruft man beispielsweise eine Auflistung aller Räume ab, so wird für jeden Raum der Endpunkt seiner Etage mitgesendet (siehe Listing oben).

Neben den Daten zum Raum enthält das Objekt auch die Adressen zu anderen Endpunkten, die den Raum betreffen. Dieses Prinzip nennt sich HATEOAS (Hypermedia As The Engine Of Application State) und ergänzt REST. Statt die Information über die verfügbaren Endpunkte aus einer API-Dokumentation auszulesen, werden Sie direkt über das API angeboten und machen dem Programmierer das Leben leichter. Gleichzeitig werden Applikationen, die das API anzapfen, dadurch robuster und weniger fehleranfällig, wenn sich das API ändert. Außerdem werden Aktionen nur dann angeboten, wenn sie auch verfügbar sind. Ist ein Raum nicht buchbar, wird auch der Link `book` nicht angeboten.

Auch schreibende Zugriffe auf das API, beispielsweise die Erstellung eines neuen Raums, funktionieren per JSON. Der Client sendet dem Server ein JSON-Objekt mit den Informationen des neuen Raums und bekommt eine JSON-Antwort zurück. Wurde beispielsweise ein zu kurzer Name oder eine negative Sitzanzahl angegeben, würde sich der Server mit einer Fehlermeldung mit entsprechendem HTTP-Code beschweren. Wenn die Erstellung erfolgreich ist, fällt die Antwort recht kurz aus:

```
{
  "meta": {
    "status": 201,
    "request_ms": 198
  },
  "payload": {
   "success": true,
   "id": 4,
   "links": [
   {"rel": "room", "href": "/rooms/4"}
   ]
  }
}
```

Sie enthält neben den üblichen Metadaten und dem weiterführenden Link auch die ID des neu erstellten Raums. Somit kann der API-Client direkt weiterarbeiten.

Der Kunde

Um auf eine beliebige REST-API zugreifen zu können, muss man keine besonderen Kunststücke vollführen. Jede Software, die in der Lage ist, HTTP-Anfragen an einen Server zu senden, kann auch Daten von einem REST-API anfragen. Die denkbar einfachste Form ist dabei der `curl`-Befehl in der Kommandozeile (neuerdings auch unter Windows). So können Sie beispielsweise GET-Anfragen an die REST-API Attrappe „JSONPlaceholder“ senden (siehe ct.de/yuz6) und die Antwort lesen:

```
curl https://jsonplaceholder.↵
↳typicode.com/posts/
```

Der Aufruf über `curl` ist nicht nur für Experimente nützlich. In der Praxis werden REST-APIs so über Skripte oder Cronjobs aufgerufen, um zum Beispiel automatisiert Daten zu synchronisieren oder Logs zu löschen.

REST-APIs sind nicht nur sinnvoll, wenn man Daten von externen Quellen bezieht. Es kann auch sinnvoll sein, die eigene Seite in Frontend und Backend zu teilen und Daten per API-Aufruf mit JavaScript zu laden. Backend und Frontend stehen dann zueinander wie ein klassisches Autoradio zu seiner abnehmbaren Bedienblende.

Sie speichert keine Daten, sondern dient ausschließlich der Anzeige und gibt Befehle ans Backend weiter. Über die standardisierte Schnittstelle kann zum nächsten Sender gewechselt werden. Welche Blende dieses Befehle sendet, ist dem Radio egal – solange die Blende mit der Schnittstelle kompatibel ist, kann sie Daten senden und auslesen.

Genauso ist es einem REST-API egal, ob die HTTP-Requests über die Webanwendung kommen, über die Smartphone-App oder einen Alexa-Skill. Die Vorteile liegen auf der Hand: Die Geschäftslogik einer Anwendung muss nur ein einziges Mal entwickelt werden – völlig unabhängig von den Betriebssystemen oder Endgeräten. Eine Versicherung könnte beispielsweise ein API entwerfen, auf das der Kundenberater unterwegs mit seiner App zugreift, während der Sachbearbeiter intern mit einem Desktop-Programm arbeitet. Der Kunde nutzt eine Weboberfläche

oder baut sich anhand der API-Dokumentation selbst eine Alexa-Integration.

Den dritten Grund, REST-APIs zu entwickeln, zeigen Vergleichsplattformen: Die kostenlose Bereitstellung von Daten an Externe ist beispielsweise für Fluggesellschaften und Hotels überlebenswichtig, um bei Vergleichsportalen gelistet werden zu können. Datenbereitstellung über ein API ist aber auch zu einem Geschäftsmodell geworden. Viele API-Anbieter sehen sich als Dienstleister und verlangen teilweise kleine Centbeträge pro Aufruf – eine gute Übersicht über kostenlose und kostenpflichtige öffentliche APIs für verschiedene Themenbereiche liefert die Webseite ProgrammableWeb (ct.de/yuz6).

In der einfachsten Form kann ein REST-API auch über die Kommandozeile angezapft werden.

```
Eingabeaufforderung
}
C:\Users\ct>curl https://jsonplaceholder.typicode.com/users
[
  {
    "id": 1,
    "name": "Leanne Graham",
    "username": "Bret",
    "email": "Sincere@april.biz",
    "address": {
      "street": "Kulas Light",
      "suite": "Apt. 556",
      "city": "Gwenborough",
      "zipcode": "92998-3874",
      "geo": {
        "lat": "-37.3159",
        "lng": "81.1496"
      }
    },
    "phone": "1-770-736-8031 x56442",
    "website": "hildegard.org",
    "company": {
      "name": "Romaguera-Crona",
      "catchPhrase": "Multi-layered client-server neural-net",
      "bs": "harness real-time e-markets"
    }
  },
  {
    "id": 2,
    "name": "Ervin Howell",
    "username": "Antonette",
    "email": "Shanna@melissa.tv",
    "address": {
      "street": "Victor Plains",
```

Die Essensreste

Zurück ins Restaurant: Wenn der Kunde sein Gericht gegessen hat, lässt er vielleicht die Oliven aus dem Salat übrig. Sie hätten gar nicht ausgeliefert werden müssen. Ähnliches passiert auch bei API-Aufrufen. Will man im Raumplan-Beispiel nur Raumnummer, Ebene und Sitzplätze auflisten, muss man alle Informationen anfragen. Die Beschreibungen eines jeden Raums, seine Größe und seine weiterführenden Links werden mitversandt, bleiben aber ungenutzt liegen. Die nicht benötigten versandten Daten müssen durchs Internet transportiert werden, kosten Rechenzeit und verlangsamen die Antwort. Praktischer wäre es, wenn der Kunde die Bestellung „ohne Oliven“, also für das Beispiel der Raumverwaltung „ohne Beschreibungen“ aufgeben könnte.

Um dieses Problem zu lösen, erweitern viele API-Anbieter den REST-Aufruf um Filtermöglichkeiten, damit der Client die benötigten Felder auswählen und so ressourcensparende Anfragen stellen kann. Facebook geht mit seiner Abfragesprache GraphQL noch einen deutlichen Schritt weiter. Der Client bestimmt beim Abruf, in welcher Struktur er die Antwort erhalten möchte und hat somit die volle Kontrolle über die gesendeten Daten.

(jam@ct.de) c't

***Dokumentationen:* ct.de/yuz6**